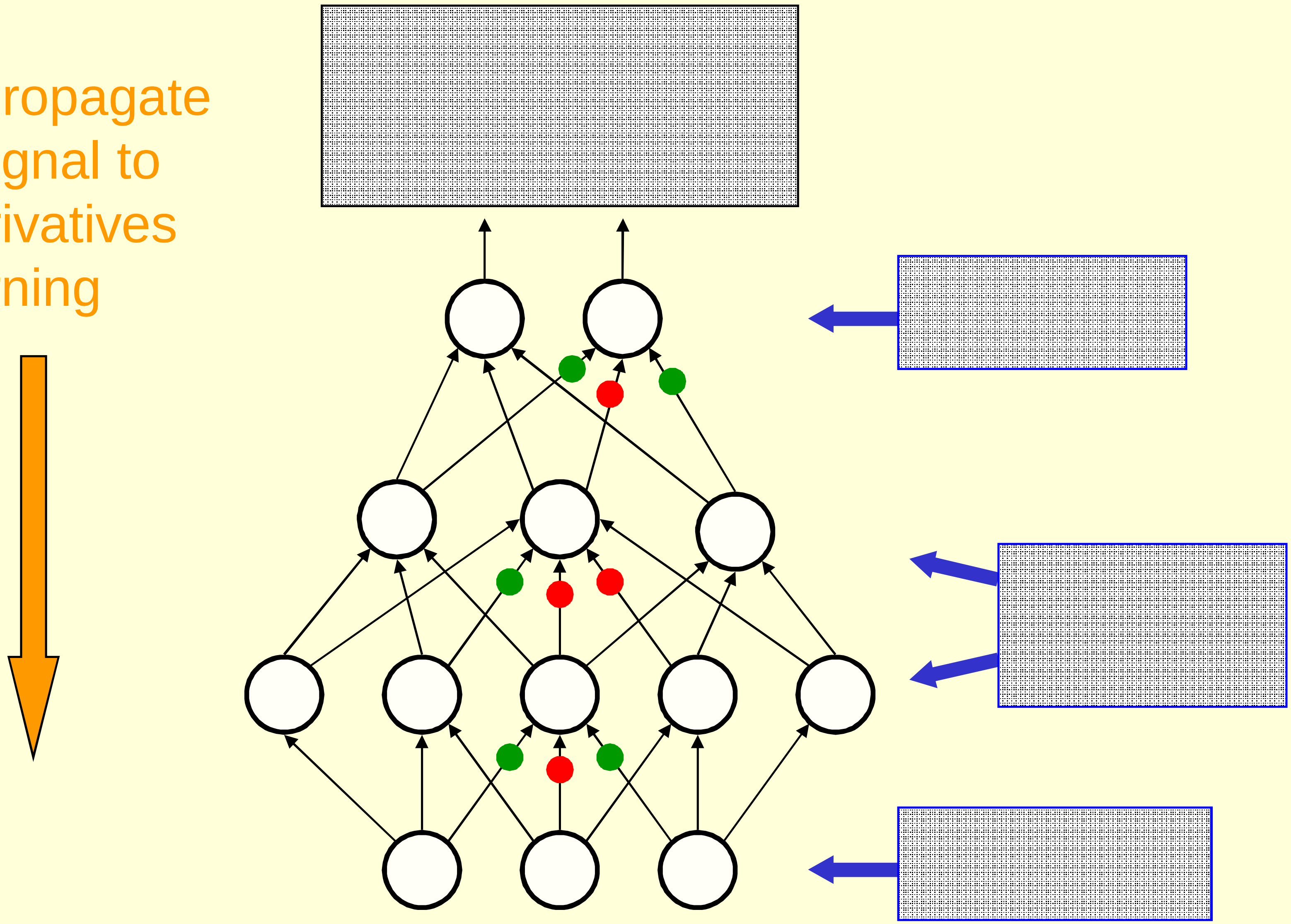
Neural Networks

Lecture 4
Learning to model relationships and word sequences

Learning by back-propagating error derivatives

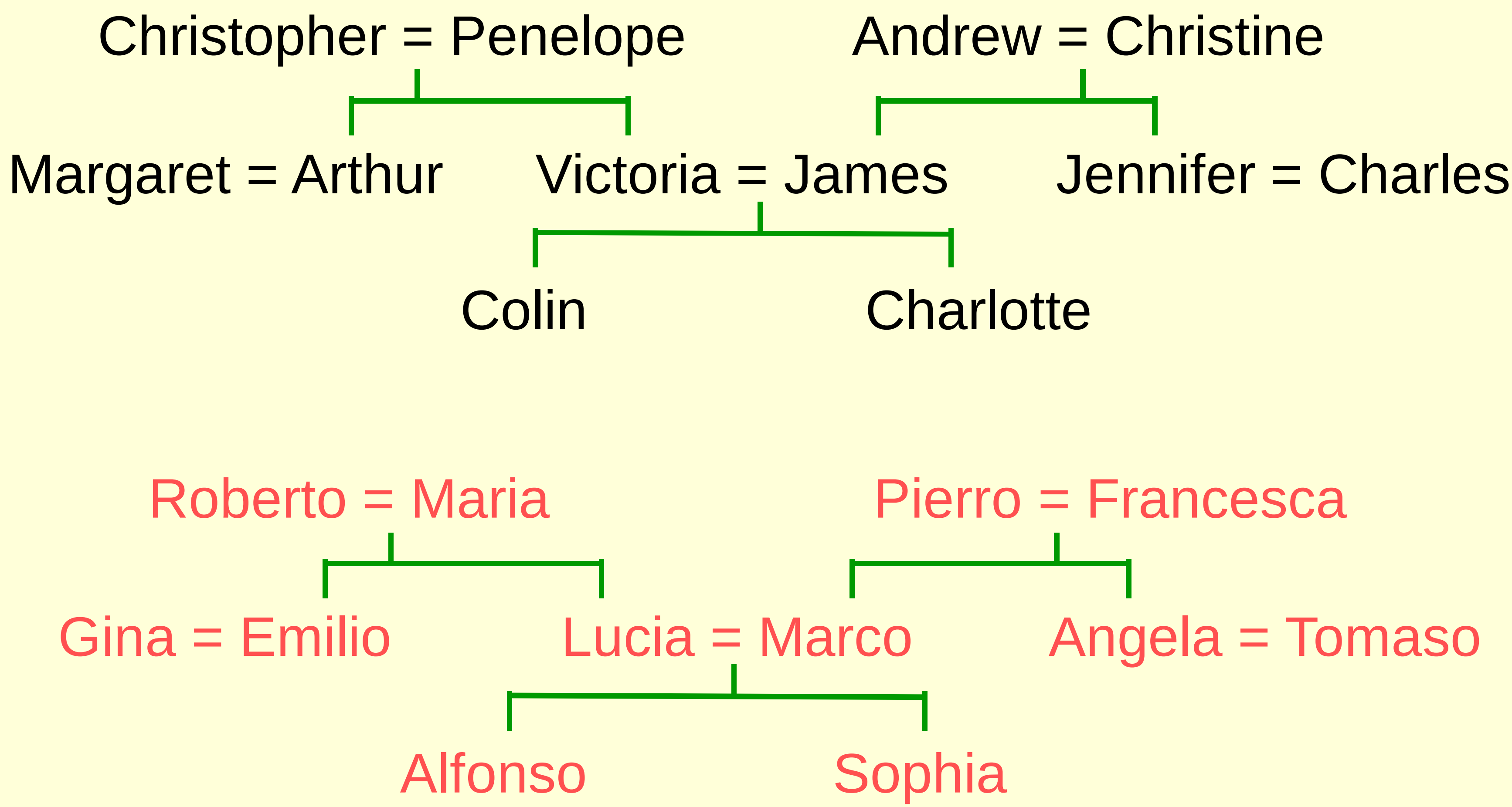
Back-propagate error signal to get derivatives for learning



Some Success Stories

- Back-propagation has been used for a large number of practical applications.
  - Recognizing hand-written characters
  - Predicting the future price of stocks
  - Detecting credit card fraud
  - Recognize speech (wreck a nice beach)
  - Predicting the next word in a sentence from the previous words
    - This is essential for good speech recognition.
  - Understanding the effects of brain damage

An example of relational information



Another way to express the same information

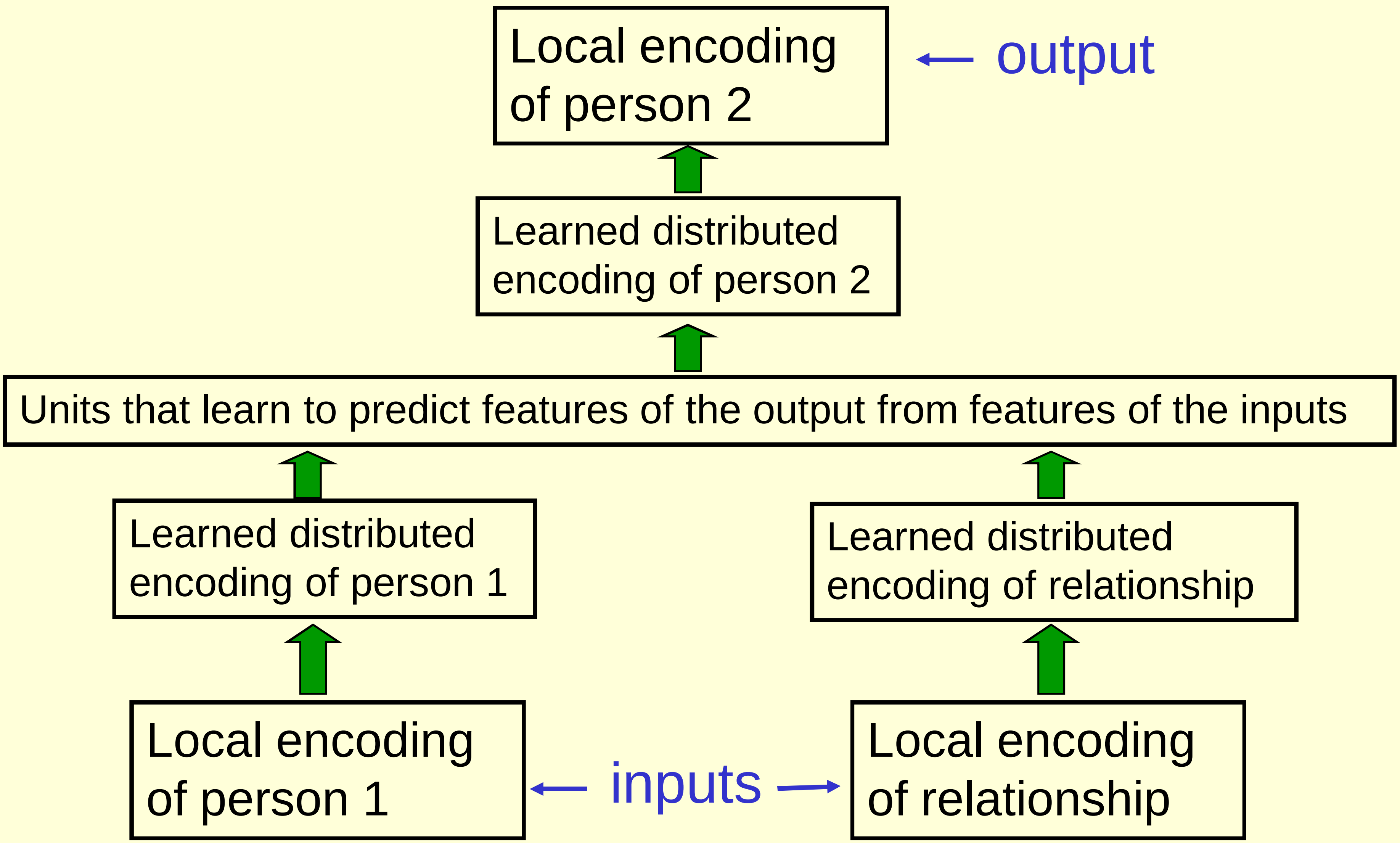
- Make a set of propositions using the 12 relationships:
  - son, daughter, nephew, niece
  - father, mother, uncle, aunt
  - brother, sister, husband, wife
- (colin has-father james)
- (colin has-mother victoria)
- (james has-wife victoria) this follows from the two above
- (charlotte has-brother colin)
- (victoria has-brother arthur)
- (charlotte has-uncle arthur) this follows from the above

A relational learning task

- Given a large set of triples that come from some family trees, figure out the regularities.
  - The obvious way to express the regularities is as symbolic rules

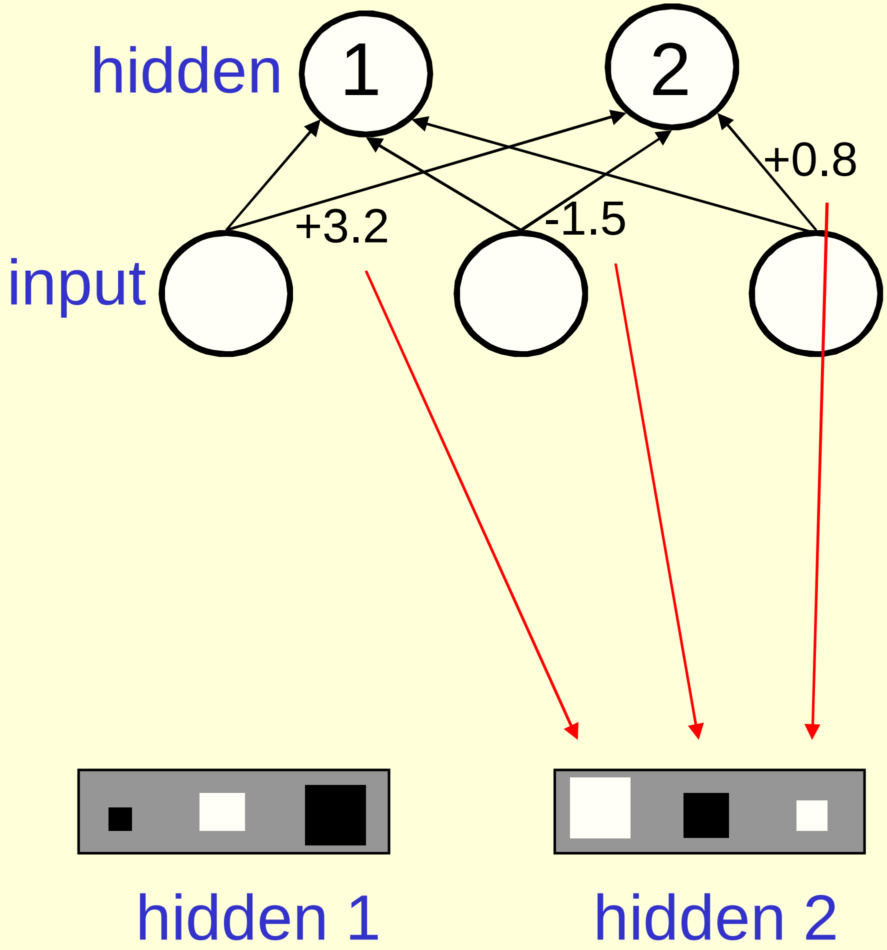
    (x has-mother y) & (y has-husband z) => (x has-father z)
- Finding the symbolic rules involves a difficult search through a very large discrete space of possibilities.
- Can a neural network capture the same knowledge by searching through a continuous space of weights?

The structure of the neural net

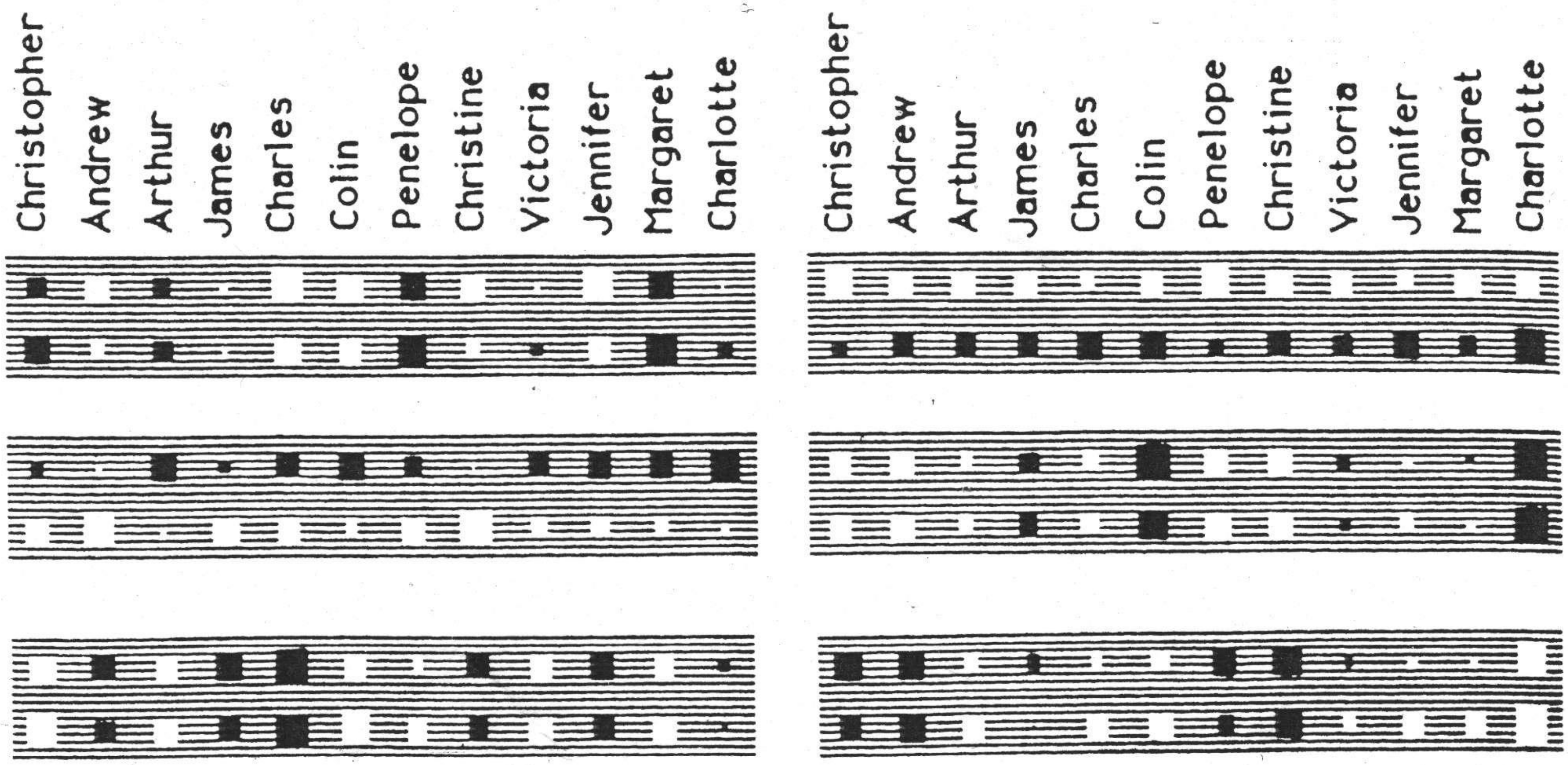


How to show the weights of hidden units

- The obvious method is to show numerical weights on the connections:
  - Try showing 25,000 weights this way!
- Its better to show the weights as black or white blobs in the locations of the neurons that they come from
  - Better use of pixels
  - Easier to see patterns



The features it learned for person 1



Christopher = Penelope   Andrew = Christine

Margaret = Arthur   Victoria = James   Jennifer = Charles

Colin   Charlotte

What the network learns

- The six hidden units in the bottleneck connected to the input representation of person 1 learn to represent features of people that are useful for predicting the answer.
  - Nationality, generation, branch of the family tree.
- These features are only useful if the other bottlenecks use similar representations and the central layer learns how features predict other features. For example:

  Input person is of generation 3 and

  relationship requires answer to be one generation up

  implies

  Output person is of generation 2

Another way to see that it works

- Train the network on all but 4 of the triples that can be made using the 12 relationships
  - It needs to sweep through the training set many times adjusting the weights slightly each time.
- Then test it on the 4 held-out cases.
  - It gets about 3/4 correct. This is good for a 24-way choice.

Why this is interesting

- There has been a big debate in cognitive science between two rival theories of what it means to know a concept:

  The feature theory: A concept is a set of semantic features.
  - This is good for explaining similarities between concepts
  - Its convenient: a concept is a vector of feature activities.

  The structuralist theory: The meaning of a concept lies in its relationships to other concepts.
  - So conceptual knowledge is best expressed as a relational graph.
- These theories need not be rivals. A neural net can use semantic features to implement the relational graph.
  - This means that no explicit inference is required to arrive at the intuitively obvious consequences of the facts that have been explicitly learned. The net "intuits" the answer!

A subtelty

- The obvious way to implement a relational graph in a neural net is to treat a neuron as a node in the graph and a connection as a binary relationship. But this will not work:
  - We need many different types of relationship
    - Connections in a neural net do not have labels.
  - We need ternary relationships as well as binary ones. e.g. (A is between B and C)
  - Its just naïve to think neurons are concepts.

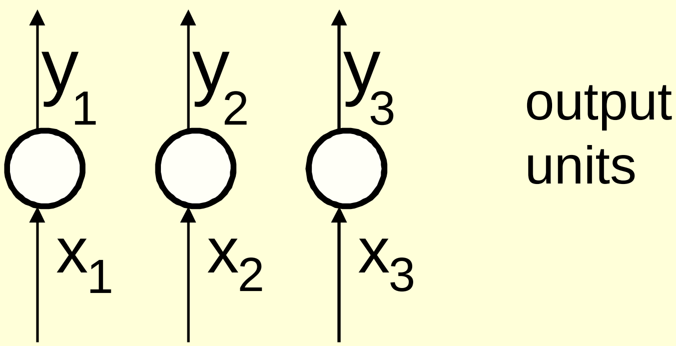
Problems with squared error

- The squared error measure has some drawbacks
  - If the desired output is 1 and the actual output is 0.00000001 there is almost no gradient for a logistic unit to fix up the error.
  - If we are trying to assign probabilities to class labels, we know that the outputs should sum to 1, but we are depriving the network of this knowledge.
- Is there a different cost function that is more appropriate and works better?
  - Force the outputs to represent a probability distribution across discrete alternatives.

Softmax

The output units use a non-local non-linearity:

$$y_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$



$$\frac{\partial y_i}{\partial x_i} = y_i \, (1 - y_i)$$

desired value

The cost function is the negative log prob of the right answer

$$C = -\sum_j d_j \log y_j$$

The steepness of C exactly balances the flatness of the output non-linearity

$$\frac{\partial C}{\partial x_i} = \sum_j \frac{\partial C}{\partial y_j} \frac{\partial y_j}{\partial x_i} = y_i - d_i$$

A basic problem in speech recognition

- We cannot identify phonemes perfectly in noisy speech
  - The acoustic input is often ambiguous: there are several different words that fit the acoustic signal equally well.
- People use their understanding of the meaning of the utterance to hear the right word.
  - We do this unconsciously
  - We are very good at it
- This means speech recognizers have to know which words are likely to come next and which are not.
  - Can this be done without full understanding?

The standard "trigram" method

- Take a huge amount of text and count the frequencies of all triples of words. Then use these frequencies to make bets on the next word in a b ?

$$\frac{p(w_3 = c \mid w_2 = b, w_1 = a)}{p(w_3 = d \mid w_2 = b, w_1 = a)} = \frac{count(abc)}{count(abd)}$$

- Until very recently this was state-of-the-art.
  - We cannot use a bigger context because there are too many quadgrams
  - We have to "back-off" to digrams when the count for a trigram is zero.
    - The probability is not zero just because we didn't see one.

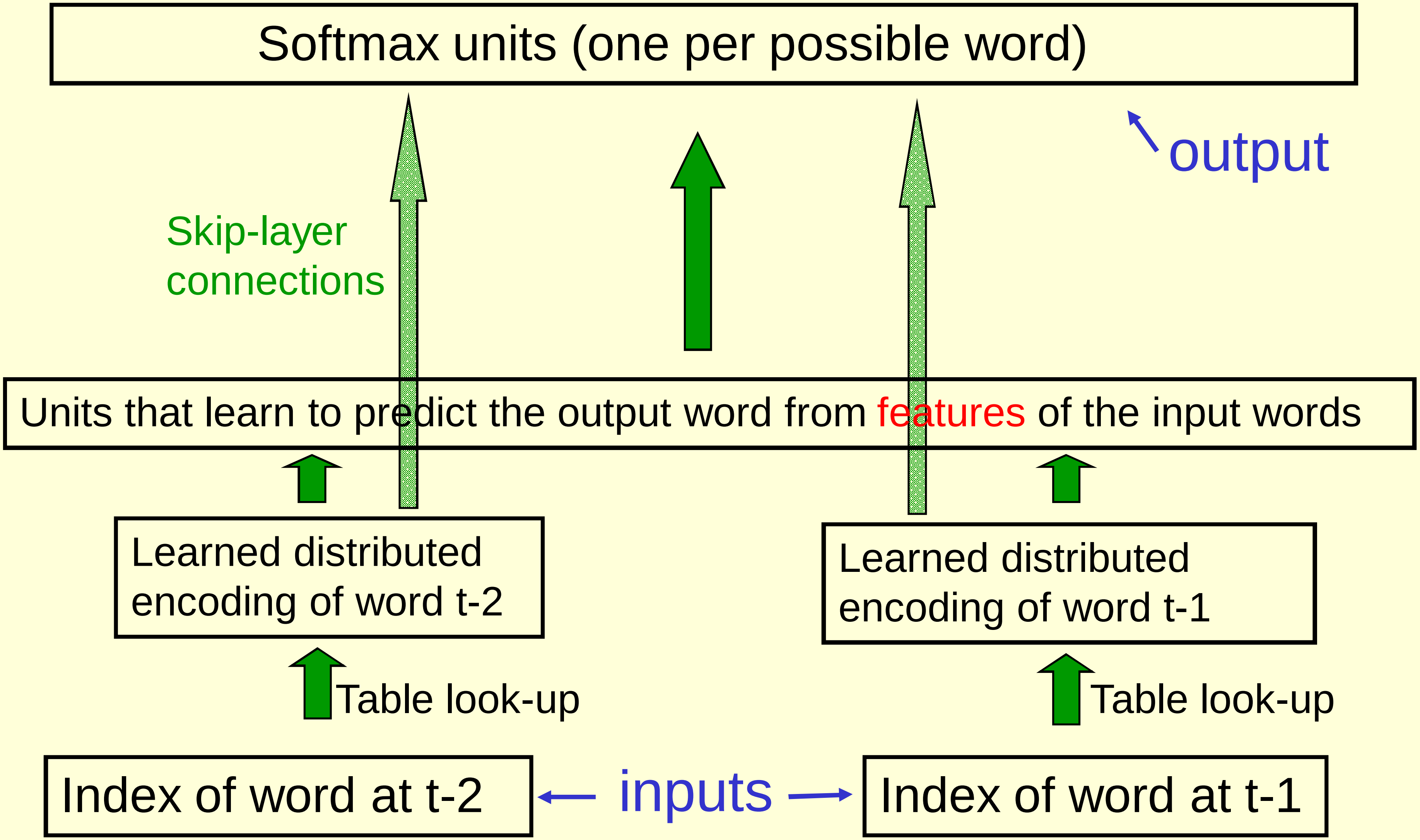
Why the trigram model is silly

- Suppose we have seen the sentence

  "the cat got squashed in the garden on friday"
- This should help us predict words in the sentence

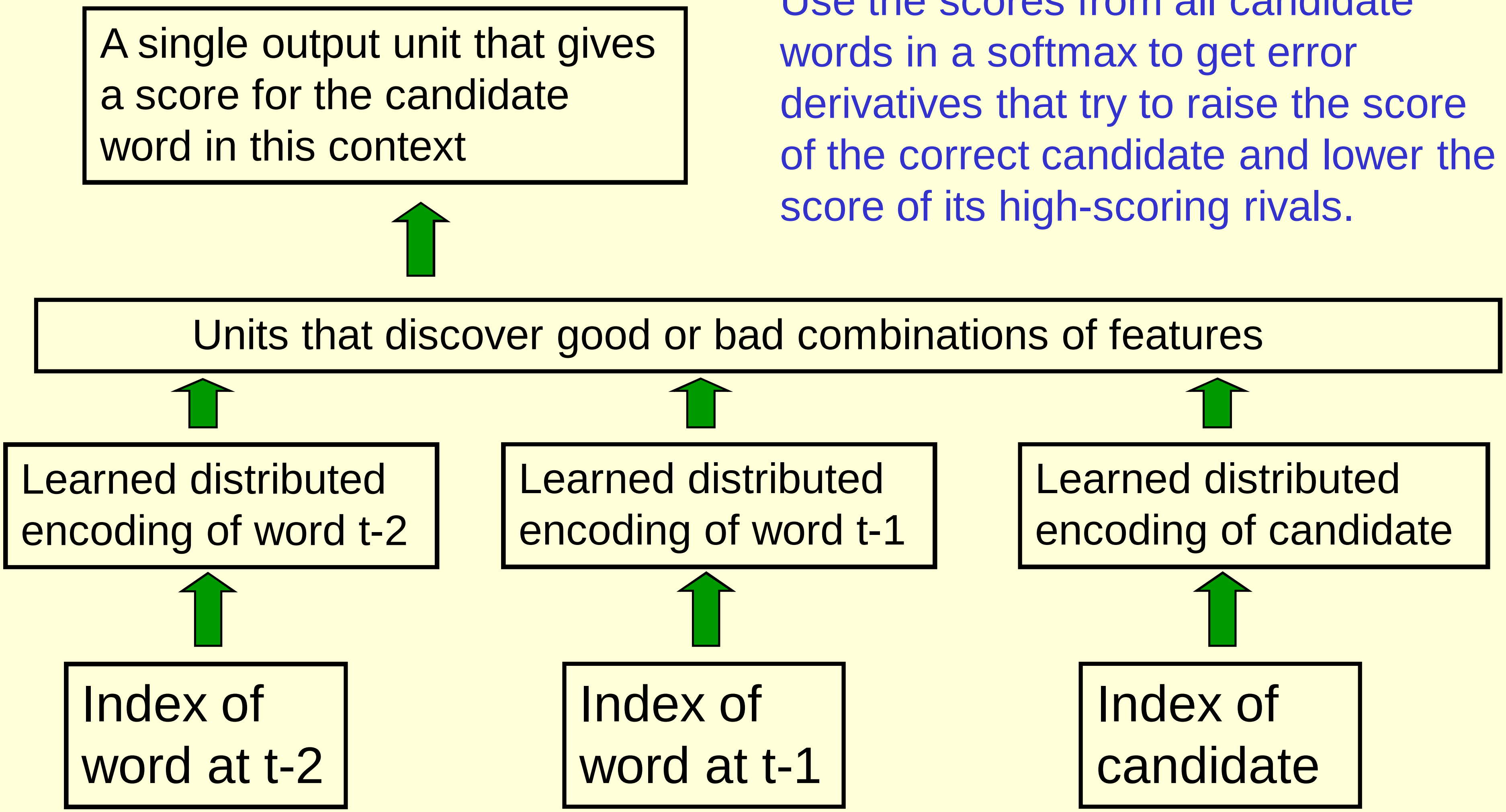
  "the dog got flattened in the yard on monday"
- A trigram model does not understand the similarities between
  - cat/dog squashed/flattened garden/yard friday/monday
- To overcome this limitation, we need to use the features of previous words to predict the features of the next word.
  - Using a feature representation and a learned model of how past features predict future ones, we can use many more words from the past history.

Bengio's neural net for predicting the next word



An alternative architecture

Use the scores from all candidate words in a softmax to get error derivatives that try to raise the score of the correct candidate and lower the score of its high-scoring rivals.



Try all candidate words one at a time

The Collobert and Weston net

- Learn to judge if a word fits the 5 word context on either side of it. Train on ~600 million words.